15th ICCRTS
The Evolution of C2

Title: "Engineering and Acquiring C4ISR Systems Based on SOA"

Paper ID: # 134

Potential Topic Areas:
Topic 1 - Concepts, Theory and Policy
Topic 7 - C2 Approaches and Organization
Topic 9 - C2 Architectures and Technologies

Authors:
Antonio Siordia (Point of Contact)
Lee Zimmerman

SPAWAR Systems Center Pacific
53560 Hull Street
San Diego, CA 92152-5001
619-553-5601
antonio.siordia@navy.mil

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2010		**00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Engineering and Acquiring C4ISR Systems Based on SOA	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **SPAWAR Systems Center Pacific,53560 Hull Street,San Diego,CA,92152-5001**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Proceedings of the 15th International Command and Control Research and Technology Symposium (ICCRTS '10), Santa Monica, CA, June 22-24, 2010

14. ABSTRACT
The United States Department of Defense (DoD) policy guidance is driving a migration towards net‐centric systems based on a service‐oriented architecture (SOA). This migration to SOA and services based systems supports the ever‐ncreasing demands for more agile C i 4ISR systems, more cost effective systems, and increased speed‐to‐capability. Individual programs of record are slowly starting to adopt elements of SOA, but little work has been done to indentify the acquisition and system‐of‐system engineering challenges that a wholesale shift to SOA entails. This paper will identify he major engineering and acquisition challenges in moving to an all SOA solution t and discuss potential solutions for many of these challenges. This paper will illustrate the challenges and potential solutions by examining real world examples from the development of key C4ISR systems ? including GCCS‐J, GCCS‐I3, DCGS, NECC and CANES.1 Parallel trends in acquisition and engineering to e explored include the move towards mission thread based requirements/testing nd platform‐level engineering.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**17**	

Engineering and Acquiring C4ISR Systems Based on SOA

Abstract

The United States Department of Defense (DoD) policy guidance is driving a migration towards net-centric systems based on a service-oriented architecture (SOA). This migration to SOA and services based systems supports the ever-increasing demands for more agile C4ISR systems, more cost effective systems, and increased speed-to-capability.

Individual programs of record are slowly starting to adopt elements of SOA, but little work has been done to indentify the acquisition and system-of-system engineering challenges that a wholesale shift to SOA entails. This paper will identify the major engineering and acquisition challenges in moving to an all SOA solution and discuss potential solutions for many of these challenges.

This paper will illustrate the challenges and potential solutions by examining real world examples from the development of key C4ISR systems – including GCCS-J, GCCS-I3, DCGS, NECC and CANES.[1] Parallel trends in acquisition and engineering to be explored include the move towards mission thread based requirements/testing and platform-level engineering.

Introduction

Among the many trends and reform initiatives being applied to the acquisition of C4ISR systems today, there are three that are of particular interest in context of the move to net-centric systems based on SOA:

- The move to services-based systems
- The move to mission threads to drive acquisition
- The rise of platform-level system-of-systems engineering (in contrast to system-level engineering).

In particular, we want to look at how all of these fit together to impact – positively or negatively – the move to widespread adoption of SOA solutions within the DoD C4ISR environment.

For the Navy's next generation afloat and ashore network infrastructure acquisition efforts – CANES and NGEN – both specify requirements in terms of the specific

[1] GCCS-J (Global Command and Control System – Joint), GCCS-I3 (Global Command and Control System – Integrated Imagery and Intelligence), DCGS (Distributed Common Ground System), NECC (Net-Enabled Command Capability), CANES (Consolidated Afloat Networks and Enterprise Services)

services to be provided.[2] While certainly not all services are SOA services – for example providing Help Desk service is not a SOA service – having requirements specified as services makes it relatively straightforward to map many requirements directly to SOA services[3]. This approach also starts to drive systems to realize the benefits of SOA by characterizing the system in terms of services. These service-level requirements are often more detailed than traditional functional requirements, and by identifying common services one can begin the task of reusing them across business processes or mission threads. Today, acquisition programs typically are still executing in terms of a specific system defined as a collection of wholly owned services, but this lays the groundwork for a follow-on step of acquiring collections of services provided by multiple programs/providers (more about this later in the challenges section). The increasing importance of a services approach is reinforced by the release of the Information Technology Infrastructure Library version 3 (ITIL v3), which focuses on service management, and by the extensions to the DoD Architecture Framework (DoDAF) version 2.0 that help specify services at an architecture level.

An equally beneficial trend is towards specifying system or platform requirements in terms of what industry would call "business processes" and we in DoD call "mission threads." This recognizes that virtually no operational activity ever takes place limited to the confines of a single system. For example, carrying out a time critical strike on a ground target based on real-time data from a UAV involves many different systems – at a minimum the UAV, the UAV ground station, and a C2 system – but this thread could also include a separate imagery analysis system, a targeting system and a strike aircraft, among others. If we specify, design, and buy systems based on a set of functional requirements just for that system, we run the danger of not having systems work seamlessly together to execute even fully anticipated mission threads, let alone emergent mission requirements. The general industry approach to implementing SOA is to identify the set of services that is necessary to execute a business process – what's known in SOA terminology as "composing" services. Therefore, the trend towards having higher-level requirements specified in terms of mission threads also helps further the DoD "SOA revolution." However, just as in the discussion on defining requirements as services, the challenge faced by the

---

[2] Descriptions of referenced systems are provided in Appendix 1.

[3] The Organization for the Advancement of Structured Information Standards (OASIS) Reference Model for Service Oriented Architecture defines a service as "a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description.

acquisition community is how to buy a set of services that supports a mission thread. This will be addressed in the following topic.

The third acquisition trend of interest is probably more accurately characterized as an engineering trend of interest – the recognition that in addition to doing system-level engineering we have to develop processes and skills for system-of-system engineering. In our case, we are concerned with the design, development and integration of multiple systems into platforms, where a platform is typically a ship or a fixed or mobile command center.[4] As an illustration, the Navy Maritime Operations Center (MOC) effort, designed to bring commonality to US Navy numbered fleet command centers, still involves over 30 different C4ISR systems running on at least four different networks.

How Do You Buy a Services-based System?

Let's start with a story. A given program, Alpha, might develop a service for its own use and would logically scale the supporting infrastructure to an appropriate, relatively low demand. The program Alpha developers followed net-centric guidelines and expose the existence of their new service(s) in a directory where users associated with program Bravo discover it and find it very useful. As a result, usage of the service suddenly increases by two or three-fold and the supporting infrastructure might be overwhelmed. With current acquisition practices, both Alpha and Bravo are out of luck because the service was built to meet the requirements of the former, and there is no funding to upgrade the service infrastructure to support the users from the latter. Also, since this was not a planned dependency, there isn't any service level agreement (SLA) between programs Alpha and Bravo, and it follows that there is no guarantee that the service won't go away or change its interface (e.g. its API) or its data format.[5] In a vacuum this re-use of extant services would be considered good news, but only if the acquisition system can be modified to be able to react by shifting resources to program Alpha and to put in place an SLA to recognize the excellent service reuse and the new inter-program dependency.

Actually, even this illustration is too limited because it still speaks in terms of programs. Ultimately the goal should be for lots of organizations to develop

---

[4] For the purposes of this paper, we will be address the physical instantiation of a platform while acknowledging the inherent node-like nature of these platforms in a net-centric environment.

[5] A Service Level Agreement is typically a written document that captures technical aspects of the performance and interoperability requirements between services.

interoperable services and for an integrating organization to assemble a collection of services, assembled into workflows that meet the combined mission requirements for a particular platform. Figure 1 provides a simplified illustration of this decomposition of "stove pipe" systems into collections of shared services. In this vision, the services become the key unit of functionality instead of systems and that has huge implications for how we define, buy, test, accredit and field capabilities.

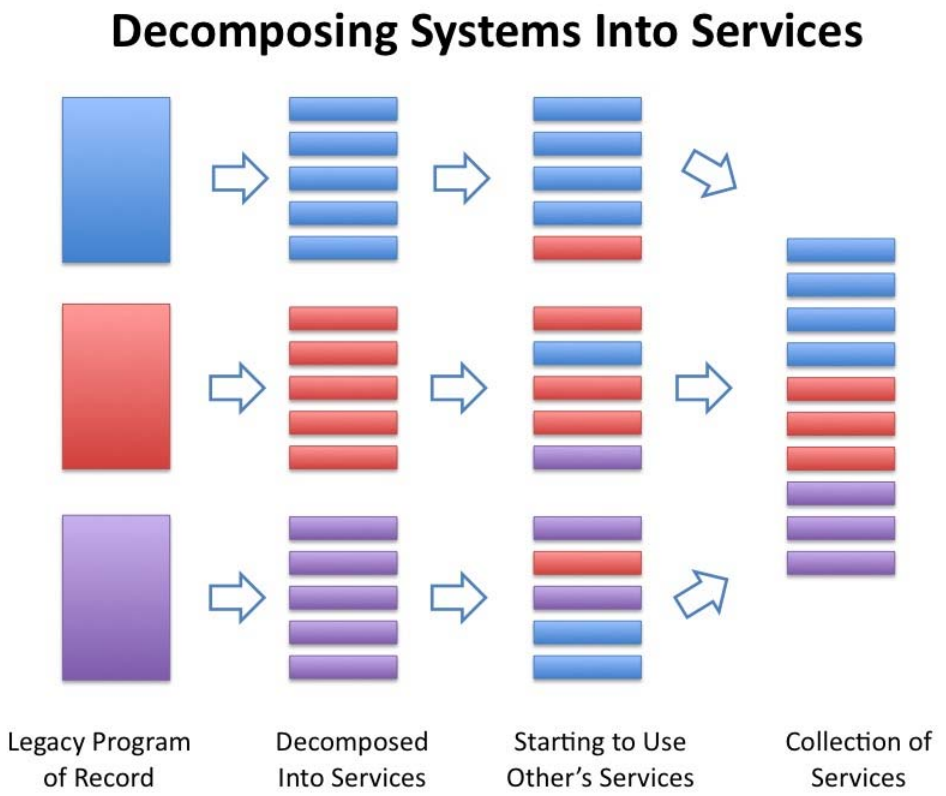

Figure 1: Decomposing Systems Into Services

In this approach technical, schedule and cost risks of shift from relatively self-contained programs of record to solutions composed on services developed by multiple programs and organizations. A project manager (PM) or acquisition program manager (APM) is not likely to be thrilled about having his or her success be wholly, or even in part, dependent on how well other programs execute their service(s) development. Instead, they may prefer to have their resource sponsors fund them to supply individual services rather than whole systems. But if that occurs, it raises the question of who is responsible for integrating services into a collection of services that actually meet a mission need?

Today, we integrate applications into systems and then systems into platforms. In the near future this model could conceivably shift to integrating services directly

into platforms and bypassing the system level altogether. Even if the acquisition community does not take a step that extreme, you still have the question of how to partition requirements to services and then services to developers. Once you've partitioned the requirements to the service level, you then have to distribute funding appropriately. Further, as the story the opening this section illustrates, funding decisions are not just limited to the development phase, but have to be revisited throughout the lifecycle of each service.[6]

The NECC program took one approach to allocating requirements and funding. Funding for the Services' (i.e. Air Force, Army, Marine Corps, and Navy) program of record (POR) C2 systems was shifted to DISA to meet an agreed upon set of Joint requirements. DISA, in turn, partitioned the money back out to the Services to develop sub-elements of the overall solution. Unfortunately, a perceived loss of both control and funding by the Services led to significant "push back" against NECC Program.[7] The GCCS-I^3 program illustrates a more successful approach where a collection of legacy programs retained their own funding but were pushed to abstract their data services apart from their application services so that multiple application services could use the same shared data services. In addition, a common user facing service was put in place so that, to the end user, GCCS-I^3 appears to be a tightly integrated system when, from an architectural perspective, it is fairly loosely coupled. The CANES program is taking another approach – all hardware funding shifted from individual PORs to CANES but the PORs retained funding and responsibility for software development. CANES provides enterprise services that the individual PORs have been adopting, either voluntarily or by mandate, and drives schedule alignment.

Once requirements and funding are allocated to multiple services, there are still two other major aspects of development that have to be coordinated/controlled – schedule and technical standards. In an ideal world, schedule alignment for services would not be an issue. Services would be seamlessly interoperable, could undergo continuous development and, if we need to physically host a service, we could just implement the most recent version of the service. If we do not need to actually host the service ourselves, we do not even need to do that, we just link the desired service into our workflows. Someday, maybe, this will be true. However, in the foreseeable future integrators of services at the system or platform level are still going to need to worry about interoperability and capability issues for each service that are tied to specific versions/releases. To address this integrators will need to have some forcing function to align development schedules. The CANES program, for example, works on fixed six-month development cycles with multiple checkpoints along the way and a graduation exercise at the end of the cycle. Services

---

[6] For a detailed overview of the life cycle of a system that the U.S. Department of Defense uses, please see the *Defense Acquisition Guidebook*, DAU, January 2010. < https://dag.dau.mil>

[7] Based on personal experience with the NECC program and discussions with various DISA personnel.

that fail any of these events do not get fielded as part of that baseline and fall back to the next release. This makes sense, as long as the services are all properly decoupled and backwards compatible so that an older version of one service doesn't fail to work with the newer versions of other services.

That potential failure of services to work together is what we hope to prevent through the use of appropriate technical standards. For the most part, we do not care what goes on inside a service, which gives developers considerable creative and technical latitude. We do, however, care very much about the interactions *between* services, so that well-defined, rigidly enforced standards for data formats, inter-service communication and the use of common services (like security services) are critically important.

The GCCS-I^3 program took an interesting approach to this problem by mandating the use of common database (i.e. MIDB) as the single data store for many of its component applications. This solved interoperability and data replication issues because each service or application only had to talk to the database and the data formats and API for that were very mature and well documented. Since all the data for the system is in this single, admittedly large, database multiple instances of I^3 can be kept in sync by using the built-in database replication capabilities. Most other SOA implementations take a more traditional approach by defining XML data standards, how services "expose" themselves, and how they communicate.

How Do You Test a Services-based System?

The fundamental challenge when it comes to testing the services based systems discussed here, is the realization that it's not practical to test all possible combinations of all services. While every service cannot be used in combination with any other service, this still approaches an N-factorial problem in terms of combinations and permutations of possible service compositions. One way to address the challenge is to identify the critical mission threads for any given "system" (i.e. a deployed collection of services) and make sure you test all of the service compositions that support those mission threads. This has the advantage of making sure that what must work works. It should also exercise almost all of the services, in at least one workflow, because the services we should be fielding are those that support the key missions. This would be a good platform- or organization-specific approach.

Another approach that's more generalized would be to test lots of mission threads against a collection of services to create an approved baseline of services that are known to work together. This might make sense for a program like CANES because not every ship is expected to execute all possible naval missions, but it would be markedly more efficient not to do a separate test event for every different platform's specific collection of services. This is similar to old Global Command and Control System-Maritime (GCCS-M) segment approach where all GCCS-M segments are tested, but every ship doesn't get all the segments.

With either of these two approaches, the test environment will present a challenge due to the sheer scope of the services required. Again, using CANES as an example, there are Navy-specific enterprise services running locally on the ship but these need to be able to hand-off user credentials and requests for information to services that are off the ship (for example, through NCES services). So to fully test CANES you need the full CANES baseline (ideally spread across representations of several platforms), the Naval telecommunications and Global Information Grid (GIG) infrastructure that provides the cross-platform connection and all the other services (external to CANES) that support the mission threads under test. This is the reason that the Space and Naval Warfare Systems Command (SPAWAR) and Program Executive Office Command, Control, Communication, Computers and Intelligence (PEO C^4I) have developed the Enterprise Engineering and Certification (E2C) lab and process, to provide the very large scale, distributed, end-to-end environment to support his type of testing.

A key element of that approach is to get the certifying agencies involved in the planning of test events so that a single large scale end-to-end test can be performed to serve as both the development test and the interoperability and, perhaps, operational acceptance tests. This is supported by having the certification agencies involved in defining graduation criteria as services move up through E2C "platters", as well as having them ensure that the test environment includes the platforms, mission threads and services necessary to meet their test requirements. The platter concept, first championed by the NECC Federated Development and Certification Environment (FDCE) approach to testing, involves equivalent test environments at varying levels of maturity – for example an Experimentation platter, a Developmental platter and a Production platter.

Another demand on the test environment is the level of instrumentation or sensors that will be required for SOA testing. We need to be able to collect the data needed to validate and verify Key Performance Parameters (KPPs) and other performance criteria – both at the individual service level and across the entire mission thread workflow. Equally important, there is the potential for much "finger pointing" in a SOA environment, so the test process has to be sufficiently fine-grained to identify where in the service chain a problem occurred. This is illustrated by the following example: A mission thread segment (subset of a full mission thread) is supported by a collection of five application and data services, plus associated enterprise services for security, messaging, and alerting. Each of these services have their own KPPs that have already been validated. However, when the composed workflow is executed in a realistic environment, it does so too slowly to meet the overall execution time KPP for the mission thread segment. How can that be? Perhaps each service was on the high end of it's acceptable time to execute and added together they take too long, or maybe the overhead associated with the security services across the full collection of transactions took too long? If a service is taking too long, is it due to bad design or is the hosting hardware overloaded? These are the types of questions the test environment has to be able to answer.

This baseline, end-to-end, mission thread based test approach is workable, but it does not answer the speed-to-capability requirement that is one of the selling points of a SOA approach. To achieve speed-to-capability, we also need the ability to test an updated version of a single service. This could be accomplished by plugging the updated service into the full baseline testing environment and rerunning the full multiple mission thread test. However, there needs to be a less resource intensive way of understanding the touch points for single service and just testing those elements of the service interaction, as well as for the correct service functionality.

How Do You Accredit a Services-based System?

DoD operates a robust certifying and accreditation (C&A) process for traditional stovepiped/siloed, vertical systems.[8] Today, however, the challenges of achieving service/system C&A in a net-centric environment range from dealing with varying levels of classification, to trust amongst various service providers (that may or may not be under your authority), to often-disjoint communities of interest (COI), to simply not having spent decades gaining experience in C&A as the DoD has with more traditional, discrete information systems.[9] Crossing boundaries between organizations has to be transparent to the user. , This is a key interoperability requirement that was much easier to address in the "good old days" when we built point-to-point interfaces that didn't need to worry about security. Moreover, a foundational concept of net-centricity – exposing data and services for anyone to discover and use – can help the opposing force know what is available that they might otherwise have missed when information systems weren't so intentionally exposed.

In addition, the modular, dynamic, distributed design of a SOA system goes against the underlying approach to C&A today. In a traditional system you can define the exact components that make up the system, where they are located, what version of every software element they are running and all the physical and logical interfaces. Armed with that knowledge, the security team can identify known weaknesses and verify, through testing, that vulnerabilities have been addressed. A SOA system does not have well defined boundaries because the services that make up a "system" can be hosted anywhere on any hardware running on any operating system. Worse, in a system design that allows for dynamic composition of services, we can't even

---

[8] Department of Defense. *Information Assurance Certification and Accreditation Process*. Instruction 8510.01. 28 Nov. 2007 <www.dtic.mil/whs/directives/corres/pdf/851001p.pdf>

[9] A somewhat unique aspect of DoD-related security concerns include the management of numerous levels of classification (e.g. confidential, secret, etc.). Additional concerns come into play when working with international partners, and when dealing with other U.S. and state government agencies. The most recent guidance for this is can be found in "Executive Order 13526 of December 29, 2009, *Classified National Security Information*". The National Archives. <www.archives.gov/federal-register/executive-orders/2009-obama.html#13526>

specify the full range of services that will be working together. Obviously, a completely new approach to C&A is going to be required.

If follows then that a service-based system, or amalgamation of services, can only be as secure as the most at risk component in the system. A recent IBM briefing points out that while Web 1.0 "at least had a security model," fundamentally securing SOA/Web 2.0 "is not low hanging fruit – it is hard and requires community approach to implement[ation]."[10] From the DoD's perspective, a failure in designing and implementing effective security design can result in the significant degradation and co-opting of warfighter-critical systems.[11] For the acquisition community, when trying to balance cost, schedule, and performance, Anthony Scott et al provide a set of eight guidelines that attempt to inform the discussion on how project managers might approach the challenges of SOA C&A.[12] These include "Rule 4: Plan for Future External Relationships" which touches on the fact that some upfront planning for future re-use is critical from an interoperability perspective, and "Rule 2: Embrace Risk Management, Identification, and Planning," which describes in some detail how effective risk management of unique SOA-related concerns can help mitigate and overly burdensome C&A process, which can help limit cost and schedule risks.

Much as an open society requires trust and verification, the key benefits of SOA (e.g. the sharing of resources), means that "[m]ore robust containment [is] required in the SOA world."[13] Daly is even more directive than the previously mentioned Scott article, recommending that in building a secure SOA foundation, developers seek to "add hardware to do the security critical parts that software can't," which is all the more surprising with the recent emphasis on software-based solutions. Along the same lines, it is suggested that projects seek to "abstract aspects of security enforcement into "gateway services" or "broker" nodes," essentially making security callable as a service in order to provide the ability to implement enforcement capabilities for end-to-end transactions.[14] The takeaway with regards to accreditation and security is that, for SOA applications, it is a non-trivial matter that requires substantially expanded upfront planning as well as continual re-evaluation

---

[10] Daly, Chris. "Some Thoughts on Certification and Accreditation of SOA Environments and Digital Communities," IBM. Briefing, 2009 <
https://qp.research.ibm.com/LotusQuickr/sldemos/Main.nsf/$defaultview/53B25432B974A900852575740061FB0F/$File/ChrisDalySOAandCertifAndAccreditv2.ppt>

[11] Department of Defense. *Data Sharing in a Net-Centric Department of Defense.* Directive 8320. 02. 23 Apr. 2007 <www.dtic.mil/whs/directives/corres/pdf/832002p.pdf>

[12] Scott, Anthony et al. "Certification and Accreditation of SOA Implementations: Programmatic Rules for the DoD," *CrossTalk: The Journal of Defense Software Engineering*. November/December 2009.

[13] Daly, "Some Thoughts on Certification," 2009. p72.

[14] Daly, "Some Thoughts on Certification," 2009. p72.

throughout the system lifecycle. So that even as service-based systems expand the depth and breadth of what can be used and developed collaboratively, this expansion comes at an increased cost to ensure that warfighter-critical systems are, and remain, secure.

How Do You Field a Services-based System?

Appropriately, the final issue to address is the actual fielding of a services-based system. Most of the challenges of fielding a SOA system have been addressed in previous sections. However, there are a few system delivery issues that still need to be addressed. Typically a system is a collection of software applications and a version of that system is a specific collection of specific versions of those applications. That concept can still carry forward in a SOA environment, if a single enterprise controls all of the services that make up a "system." For example, the CANES program can develop, test and accredit a baseline version of the system and field that version to some collection of platforms. That works today because most, if not all, of the services in the CANES environment are running locally and the CANES program will ensure the different versions of CANES can talk to each other. However, in a near-future distributed, Joint, net-centric environment, the more likely scenario is hundreds of different services, developed by different organizations and enterprises, both public and private, being updated continuously and being used by creative service members in ways that were not originally intended. Clearly, this requires a new approach to the concept of configuration management and release schedules that extends beyond the boundaries of any single enterprise.

The Way Ahead for C4ISR Acquisition

As the government seeks to transition from a private sector lead systems integrator (LSI) role to a government-led one, the importance of standards compliance in ever-increasing software-intensive defense systems' development is readily apparent. Just as Secretary of Defense Robert Gates noted in April of 2009 that the U.S. "requires an acquisition system that can perform with greater urgency and agility," so too do the types of systems that we acquire need to be more agile.[15]

As illustrated by Figure 2, DoD is somewhere between Step 2 and Step 4 (depending on the specific system in question) in the evolution from legacy "stove pipe" systems and a future vision of seamless, interoperable services that can be composed to address any mission requirement. The technical challenges of implementing SOA systems, while not easy, are well known. As illustrated by the cancellation of the NECC program, the programmatic challenges of implementing SOA are greater. Realizing this vision is going to require significant changes in how we:

---

[15] Carreno, Galdorisi, Hszieh and Siordia, "Rethinking Command and Control," 14th ICCRTS, June 2009, p7.

- Define requirements in terms of mission threads and services
- Allocate requirements across multiple organizations for implementation of services
- Allocate resources for sustainment of services
- Test services
- Accredit "systems" composed of services
- Sustain "systems' composed of interdependent services.

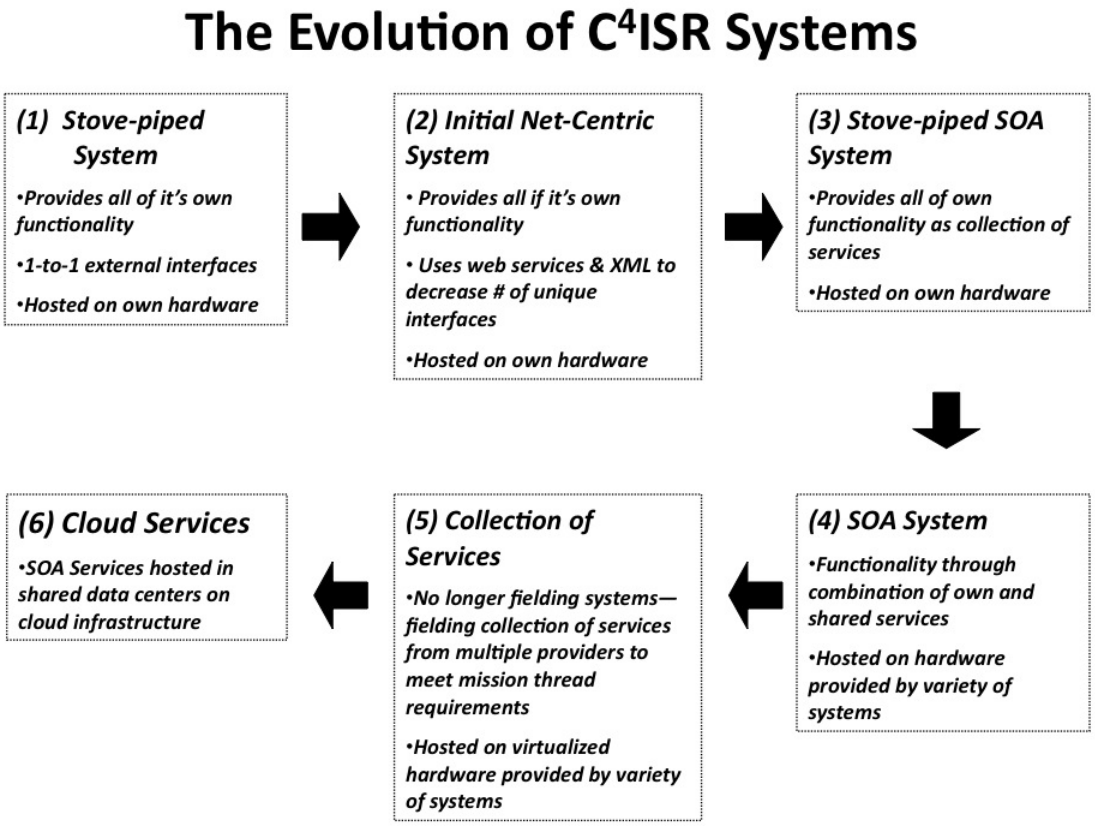

Figure 2: The Evolution of C4ISR Systems

This paper has provided an overview of the challenges faced in acquiring services based systems as we move down the evolution path for C4ISR systems. In addition, while we have not provided answers to address all of those challenges, the program examples provided has provided a starting point for developing solutions to these many challenges.

Issues for Further Investigation

In writing this paper, our understanding of the body of research on SOA and services in general was reaffirmed to be quite wide. However, its use within the acquisition of large, secure systems and platforms is a topic that could stand much additional

research. This paper's goal was to survey the host of issues, and seek to further inform the discussion in this area. Some additional research topics that would continue this line of study include:

- How do you accomplish rapid reconfiguration of a large-scale test bed to support quickly testing different services baselines?

- Is it practical to use of large-scale virtualization and simulation to put a representative end-to-end test bed "into a box." If the systems under test are virtualized, can you also virtualize the test sim/stim gear, the LAN and the WAN and still get realistic results?

- Who owns the "official" mission threads and are they sufficiently detailed to build a mission thread based test plan? Should there be a single approved library of mission threads? Could/should someone provide an approved source of consistent test data to drive mission thread testing?

- What is the formal relationship between service level KPPs and mission thread KPPs and how should this be reflecting in service requirements and testing?

- How do you factor in ad hoc compositions of services into requirements and resource decisions?

About the Authors

Lee Zimmerman is the Team SPAWAR National Competency Lead for Net-Centric Engineering and Integration. He is responsible for developing the workforce and technical processes and standards to implement platform-level and system-of-systems engineering, including those for implementing Services Oriented Architectures. With over thirty years experience in C^4I system development, Lee has worked projects that span 6.2 R&D to ILS support for operational systems and in domains as varied as national intelligence, information operations, information assurance, command & control and force protection. Lee graduated with a B.S. in Computer Science from California State University Long Beach in 1983. Lee joined

SSC SD's predecessor organization (NRAD) in 1983 and at the same time received a direct commission in the Naval Reserve where he served from 1983-1991.

Antonio Siordia works as a member of SPAWAR Systems Center Pacific's Corporate Strategy Group and is transitioning to the Net-Centric Engineering staff. He earned a Masters in Pacific International Affairs from the University of California, San Diego with a concentration in International Economics. Antonio holds several IT certifications and is currently completing a Masters in Net-Centric Systems Engineering at the Naval Postgraduate School. He joined then-SSC San Diego in 2005 as an intern, and came onboard full-time in 2007.

Appendix 1 - Program Descriptions

CANES looks to "consolidate portions of existing afloat physical networks into a reduced network infrastructure. [It] provide Afloat Core Services (ACS) to enable a flexible, agile environment for rapid, "plug-and-play" C2 and ISR capability enhancements in a Service-Oriented Architecture (SOA) environment. (Department of Defense, Space and Naval Warfare Systems Command, *Naval IT, C4ISR, Space Systems, and Enterprise Support: Today and Tomorrow* (San Diego, CA, October 2009) p57-58)

Distributed Common Ground System-Navy (DCGS-N) Program: Provides ISR and Targeting (ISR&T) system support for naval Sea Strike and Sea Basing. DCGS-N will receive and process multiple data streams from various ISR sources to provide time-critical aim points and intelligence products in support of the MOC Program. It will enhance the warfighter's COP and MDA. DCGS-N merges ISR support into a web-enabled, Net-Centric, Joint-interoperable enterprise, making ISR data visible, accessible, and understandable. As a part of the transformation to open architecture (OA) system, DCGS-N will migrate to SOA and CCE provided by CANES' ACS. (DoD, *Naval IT*, p94)

Global Command and Control System - Maritime (GCCS-M) Program: The Navy variant of the Global Command and Control System. The primary role of GCCS-M is to provide real-time (or near real-time) planning, execution and situational awareness of the battlefield environment for combatant commanders, tactical decision-makers and warfighters. The objective of the GCCS-M program is to satisfy Fleet C4I requirements through the rapid and efficient development and fielding of centrally managed C4I capability. The GCCS-M system comprises four main variants: Ashore, Afloat, Tactical/Mobile, and Multilevel Security (MLS) that together provide C2 information to warfighters in all naval environments. (DoD, *Naval IT*, p94)

Global Command and Control System – Joint (GCCS-J) enhances information superiority and supports the operational concepts of full-dimensional protection and precision engagement. It fuses select C2 capabilities into a comprehensive, interoperable system by exchanging imagery, intelligence, status of forces, and planning information. GCCS-J provides a robust and seamless C2 capability to the Combatant Commands, DoD leadership, and Service Component Commanders. GCCS-J offers vital connectivity to the systems the joint warfighter uses to plan, execute, and manage military operations. (Department of Defense, Defense Information Systems Agency, "GCCS-J," accessed Jan 11, 2009, <http://www.disa.mil/gccs-j/>)

Maritime Operations Center: Maritime Headquarters with Maritime Operation Centers (MHQ w/MOC) are the component of the United States Maritime Strategy that provide the operational strategy for fleet and when designated joint maritime operations. At present there are 11 MHQ w/MOCs planning, supporting and executing maritime military operations around the world. (*MHQ with MOC System of Systems System Engineering Plan* v11, p5)

Net-Centric Enterprise Services: Will enable net-centricity to securely interconnect people, information and capabilities, independent of time or location and provide a set of enterprise capabilities to support DoD's transformation to net-centricity. NCES will deliver these capabilities through four product lines: Collaboration, Content Discovery and Delivery (CD&D), Service-Oriented Architecture Foundation (SOAF), and User Access (Portal). NCES will provide ubiquitous access to information and services and improve interoperability to enable substantially improved planning at multiple echelons and significantly shortened decision-making cycles.

Net-Enabled Command Capability (NECC) Program: A new-start Joint C2 program led by the Defense Information Systems Agency (DISA) focused on providing commanders and warfighters with interoperable, web-enabled, timely information to make effective decisions. NECC also will provide new C2 enhancements into a fully integrated, collaborative Joint solution. NECC is DoD's principal future C2 capability that will be accessible in a net-centric environment. This approach is consistent with proposed design paths for FORCEnet, NCO, and the Global Information Grid–Enterprise Services (GIG-ES). (DoD, *Naval IT*, p94)) The NECC program was terminated as part of the Defense Authorization Bill for 2010.

Appendix B – List of Acronyms

API – Application Programming Interface
APM – Associate Program Manager
C&A – Certifications and Accreditation
C2 – Command and Control
C4ISR – Command, Control, Computer, Communications, Intelligence, Surveillance, Reconnaissance
CANES – Consolidated Afloat Networks and Enterprise Services
COI – Community of Interest
DCGS – Distributed Common Ground Systems
DISA – Defense Information Systems Agency
DoD – Department of Defense
DoDAF – Department of Defense Architecture Framework
E2C – Enterprise Engineering and Certification
FDCE – Federated Development and Certification Environment
GCCS-I³ – Global Command and Control System - Integrated Imagery and Intelligence
GCCS-J – Global Command and Control System – Joint
GCCS-M – Global Command and Control System – Maritime
GIG – Global Information Grid
ITIL – Information Technology Infrastructure Library
KPP – Key Performance Parameter
MDA – Maritime Domain Awareness
MIDB – Modernized INTEL Database
MOC – Maritime Operations Center
NCES – Net-Centric Enterprise Services
NECC – Net-Enabled Command Capability
NGEN – Next Generation Enterprise Network
PEO C⁴I – Program Executive Office, Command Control Communication, Computers and Intelligence
PM – Program Manager
POR – Program of Record
SPAWAR – Space and Naval Warfare Systems Command
SLA – Service Level Agreement
SOA – Service(s) Oriented Architecture
UAV – Unmanned Aerial Vehicle